Mpilog Quick User's Guide

1. Introduction

1.1 What is Mpilog?

Mpilog is a tool designed for the interactive generation of macromodels for the I/O ports of digital integrated circuits. The proposed macromodels match the accuracy and efficiency for the realistic system-level simulations aimed at the assessment of SI and EMC effects in high-speed digital circuits. Macromodels are mathematical relations approximating the port electrical behavior of devices, thus completely hiding the internal structure of devices and preserving the proprietary information of vendors. They are obtained from a set of port voltage and current responses carrying the information of port behavior that can be obtained either via direct measurements or via circuit simulation by driving the port with suitable stimuli. Mpilog allows the implementation of macromodels as SPICE-like subcircuits or as direct equation descriptions via metalanguages like Verilog-A. Furthermore, macromodels can be directly used in any EDA tool supporting the IBIS ver. 4.1 description of digital integrated circuits as externally defined models (see note 1 at the end of this document).

1.2 Who created Mpilog?

The original idea of describing the behavior of the I/O buffers by means of a sophisticated artificial intelligence approach came out first as a topic of a PhD thesis at Politecnico di Torino (see note 2 at the end of this document). Several developments followed inside the EMC Group of the Electronics Department, Politecnico di Torino, Torino, Italy (http://www.emc.polito.it/). A complete review of the modeling methodology is out of the scope of this document and is not reported here, but can be analyzed in greater details in references [1-8].

The Mpilog project started in 2000, under the coordination of Prof. Flavio Canavero, EMC Group leader. Dr. Igor Stievano and Ing. Claudio Siviero developed the algorithms, and Ing. Carla Giachino implemented the graphics. This project profited also of the theoretical experience of Prof. Ivan Maio.

Finally, the financial support to this project by IBM under the Faculty Award Program, is gratefully acknowledged.

1.3 About this guide

The aim of this quick guide is to provide the user with the necessary skill for a quick use of the tool for macromodel development. This document is accessible from all windows via the Help menu; in addition, short tips are displayed, in all windows, by positioning the mouse arrow on the clickable buttons.

2 Installation of Mpilog

2.1 System Requirements

Mpilog is a graphical application developed under MATLAB Version 7.1®. It is a 32-bit stand-alone application running on MATLAB-supported Microsoft Windows Systems. MATLAB license is not required to run this application (MATLAB runtime libraries are supplied with this distribution).

- *Software requirements:*
  - Windows Operating Systems: Windows XP (Service Pack 1 or 2), 2000 (Service Pack 3 or 4), NT 4.0 (Service Pack 5 or 6a).
- *Hardware requirements:*
  - Processors: Pentium III, IV, Xeon, Pentium M, AMD Athlon, Athlon XP, Athlon MP
  - Memory: 256 MB (recommended)

For more information and a complete list of supported Windows platforms, visit
http://www.mathworks.com/products/system.shtml/Windows

2.2 Licensing

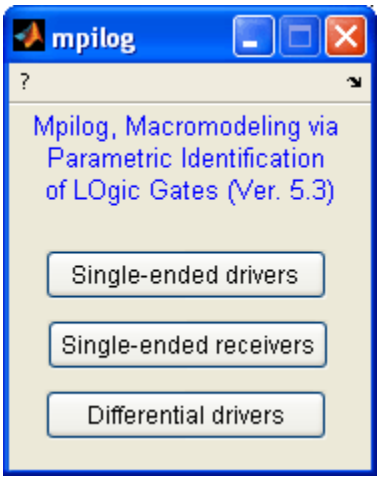
The Mpilog tool belongs to the above-named components of the EMC Group. All rights reserved. The software described in this document is provided under a license or a nondisclosure agreement, and it may be used only in accordance with terms of those agreements. No part of this documentation may be duplicated or distributed without the written permission of the tool owners.

2.3 Distribution

- The Windows-based installation package (Mpilog_X_Y.msi, where X and Y define the product version)
- The MATLAB Common Runtime library installer (MCRInstaller.exe)
- The compressed file "examples.zip"

2.4 Installing and Using Mpilog

Install the Matlab Common Runtime Libraries and Mpilog on your machine. The installation creates a shortcut on the desktop, named Mpilog X.Y, where X and Y define the product version. On launching Mpilog, the following window appears:



The button **Single-ended drivers** introduces to the generation of the macromodels for the *output and power supply ports of single-ended drivers*. The same options hold for the alternate button **Differential drivers** that provide the generation of the macromodels for differential device technologies. Similarily, button **Single-ended receivers** introduces to the generation of the macromodels for the *input and power supply ports of single-ended receivers*.

2.5 Support

Support is provided by the EMC Group. Please send inquiries to Elisasbetta Racca or Igor S. Stievano http://www.emc.polito.it/staff/

3 Tutorial on the generation of device models

The present tutorial is based on the generation of single-ended driver models, but it is sufficiently general to cover also the other devices and technologies supported by Mpilog. In fact, the modeling procedure of this tool is common for all modeling options listed in the main menu. Besides, the tool is provided with the compressed file "examples.zip" including a number of project workspaces that have already been created. The supplied workspaces, that can be easily loaded to verify the complete step-by-step modeling procedure, collect several

examples for the generation of Mpilog models of drivers and receivers belonging to single-ended and differential technologies.

The compressed file is organized as follows

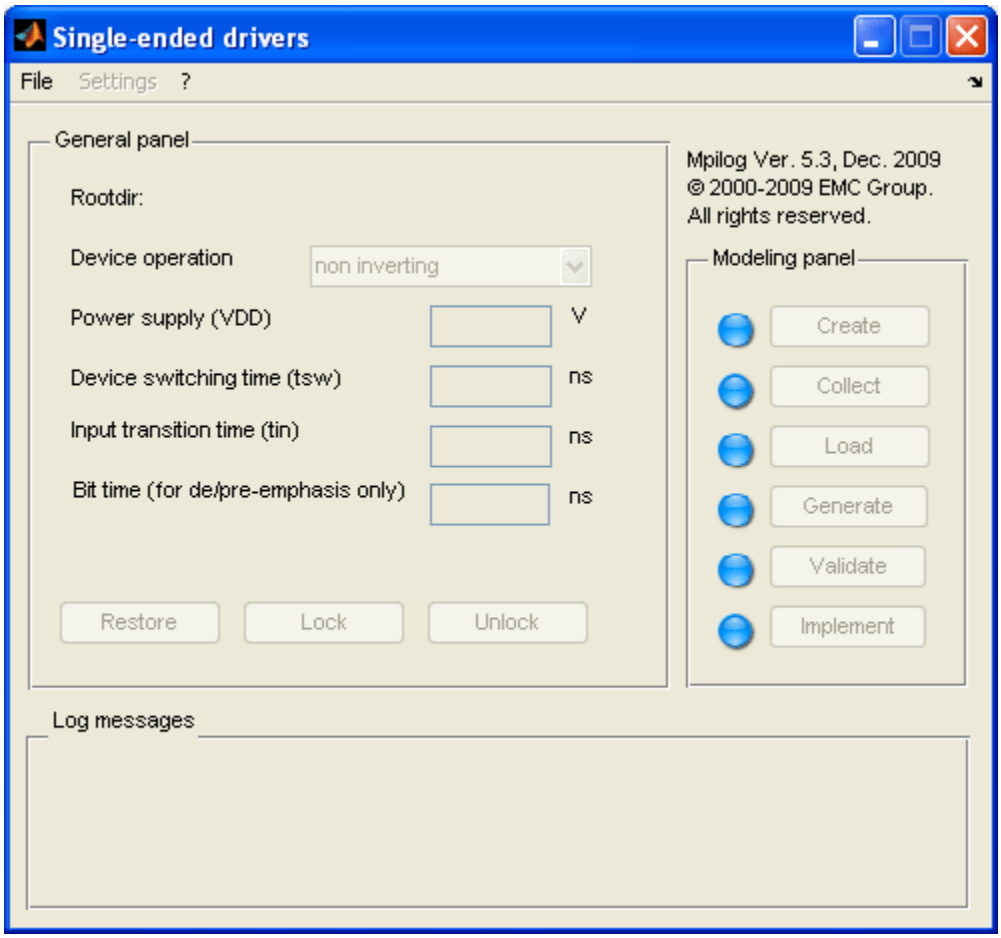
- File "examples.txt" provides the general information on the different examples that are provided.
- A set of subfolders (e.g., "sedrv_buffer") collects the Mpilog project workspaces of the different examples. Specifically, each subfolder collects the complete macromodeling tree that is automatically generated by the application of the tool to an example device. File "examples.txt" and this users' guide provide more information on the different examples.

In order to guide the reader through a complete understanding of the Mpilog macromodeling procedure, the following Section concentrates on the macromodeling of a single-ended driver. A short overview of the alternate macromodeling examples included in the example projects is detailed at the end of the document.

3.1 Example 1: project folder "sedrv_buffer"

This example refers to the application of the tool to the modeling of a simple driver circuit composed of four cascaded inverter stages, whose SPICE transistor-level description is described in [9], pag. 492. The detailed step-by-step procedure that is carried out to generate the Mpilog macromodel for this example device is reported below. The project workspace for this example is archived in the example folder "sedrv_buffer".

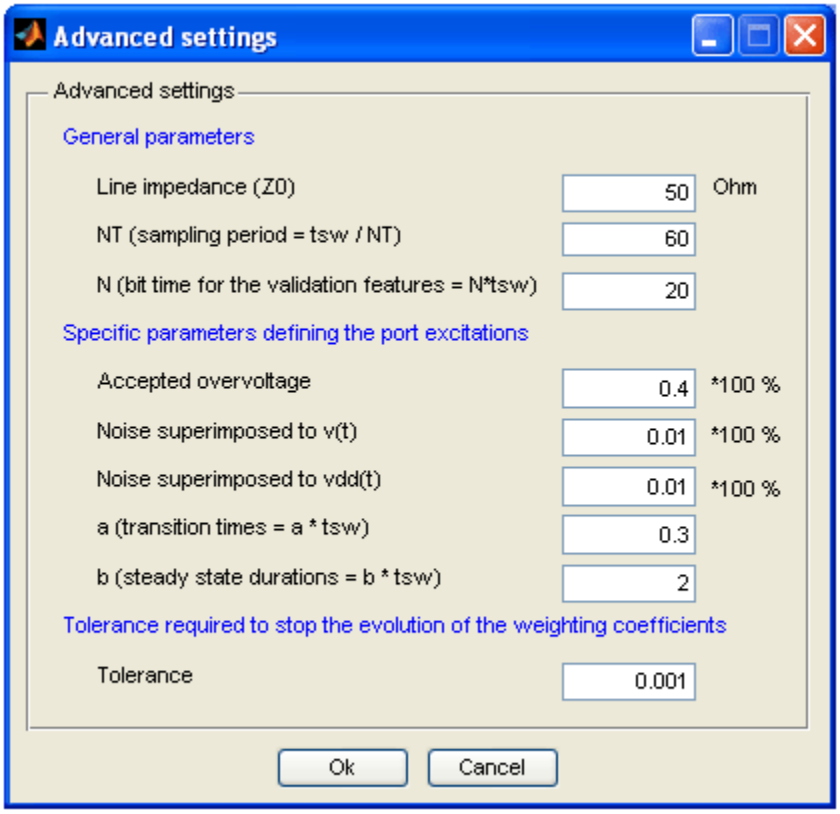
From the main Mpilog window, the initial selection of **Single-ended drivers** leads to the following window that collects the general details of the driver to be modeled (see the General panel) and the steps required by the model generation process (see the Modeling panel).



Here we suggest that the user starts to model this device by creating a new project by means of the **New** option of the drop menu **File**. The aim of this first example is to reproduce the same steps stored in the example project "sedrv_buffer". The user must specify an empty folder (e.g., "c:\temp\example1") with the purpose of collecting the project files that will be be archived in a structured manner. Such folder contains a non-editable autogenerated .MAT file ("sedrvws.MAT") saving all workspace data, and a suite of subfolders devised for a structured data file archival (see below for details). The archived workspaces of previously-used devices can be re-used, via the **Load** option of the same drop menu.

Once a new project has been created, the only general information required by the user for starting the modeling process are the **Device operation** (inverting or non inverting), the nominal values of the **Power supply** voltage, VDD, the port **Switching time**, tsw and the **Input transition time**, i.e., the rising/transition times of the input (e.g., square wave) signals applied to the input port of the driver. For this first example circuit, as suggested in the file "examples.txt", the device operation is non-inverting, VDD=5V and tsw=2ns and tin=4ns. Typically, the above values are provided by the device vendor or detailed in the device datasheet.

If needed (this choice is suggested for expert users only), some advanced parameters can be modified by using the **Advanced** link, within the settings menu, that takes the user to the following window:



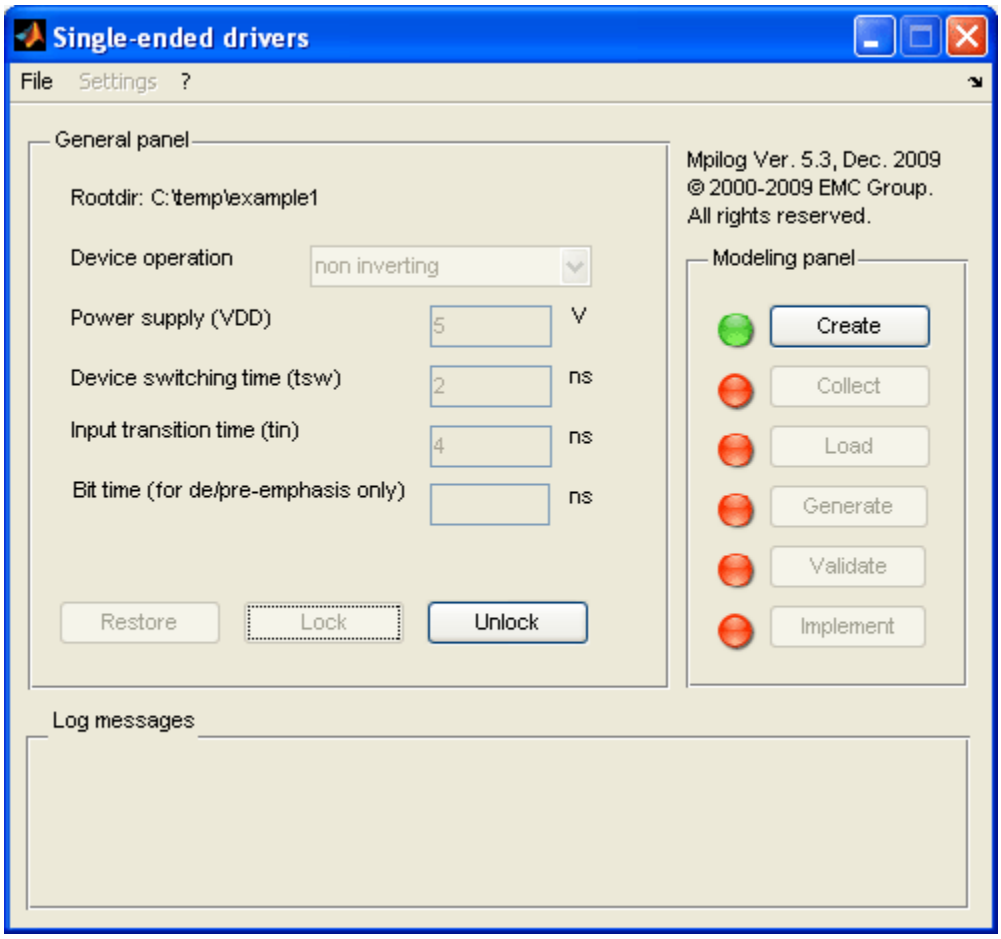
The meaning of the parameters and the possible selections in the above window are:

- **Z0** is the approximate value of the characteristic impedance of the lines connected to the device when it operates in normal condition. This value is used for both the model estimation and for the generation of a validation test consisting of the driver connected to a transmission line load.
- **NT** is used to define the sampling period used to discretize the port voltage and current responses used for the generation of the device models. The sampling period is defined as the typical port transition time (tsw) divided by NT.
- **N** is used to define the bit time used for the validation test feature included in the tool. Specifically, the bit time is computed as N*tsw.

The **remaining parameters** can be used to modify the shaping factors of the port excitations applied to the device under modeling. These excitations are required to collect the port responses used by Mpilog to compute the model parameters by fitting the model and the port responses. Since the stimuli that are commonly adopted for the identification of nonlinear dynamical systems like digital devices are multilevel signals possibly superimposed by a small amplitude noisy signal, the following parameters can be modified: *the accepted overvoltage of the multilevel signal, the ammount of noise for the excitations applied to the functional I/O ports and to the power supply port of devices, the duration of the transitions between the different lecvels of the stimuli and the duration of the flat parts (steady-states)* . The last parameter, **Tolerance**, allows the automatic computation of the duration of the weighting signals that account for the switching activity of the model and that play the same role of the input signal of a driver circuit (this parameter is for debug purpose and expert users only).

It is worth noting that the feature for macromodel validation included in Mpilog requires the installation of the HSPICE simulation program. So, in order to use the validation feature, it is recommended to locate the executable file of the HSPICE installation (hspice.exe) by using the **SPICE** button in the general panel.

Finally, when all the previous parameters are defined, the button **lock** freezes the general panel and enables the steps in the modeling panel, as shown below.



If needed, button **unlock** can be used to disable the modeling panel and to enable changes and/or modifications in general panel.

Macromodeling procedure

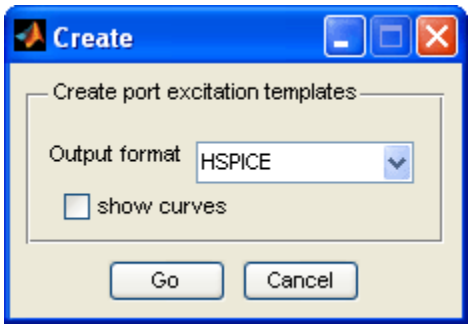
The macromodeling procedure is guided though the modeling panel, where the buttons are organized as the logical sequence of actions to be performed. Only those buttons for which the necessary data are present in the device folder, are enabled and the corresponding light is green.

In summary, the macromodeling procedure consists of the following logical steps. The first step refers to the design of the excitation signals that must be applied to the device ports in order to extract its static and dynamic behavior. The second step is the application of such stimuli to the device under modeling; this step must necessarily be performed aside from the current tool, since an experiment (real or virtual) must be performed on the device under modeling (eg, a SPICE simulation of the transistor-level description of the device). The third step is the collection of device responses. Finally, the remaining steps perform the response processing (aimed at fitting the model parameters), and subsequently validate the macromodel and generate a subcircuit implementing the same macromodel. These steps are described in more details in the following.

Step 1: Create port excitation

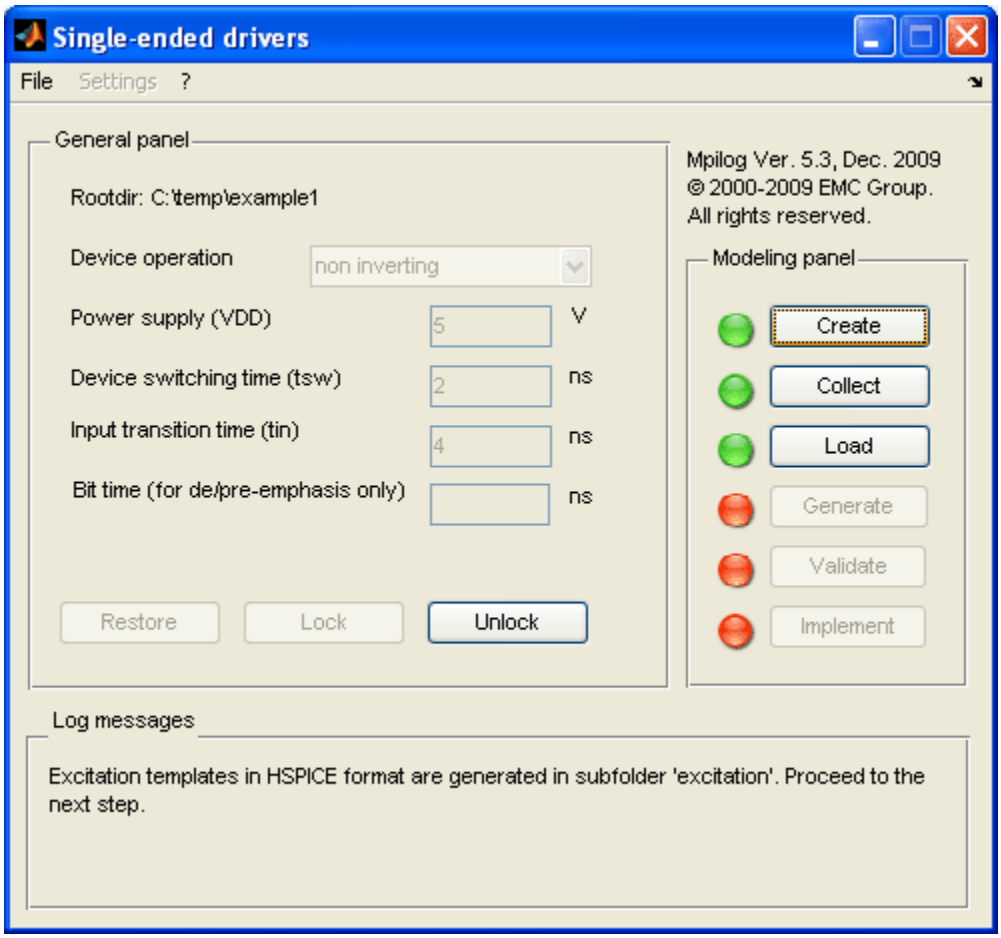
The first step of the modeling process amounts to creating simulation templates that can be emplyed by the user for stimulating the transistor-level model of the device under modeling via a suitable set of DC and transient excitations. The simulation templates automatically created by the tool will be stored in subfolder **excitation** under the modeling project tree.

The **Create** button leads to the following window:



This window allows the format of the templates (presently, conventional SPICE formats are implemented). The additional option **show curves** activates the display of the excitation signals designed by the tool; this selection is for debugging purpose, therefore it is recommended to activate this choice only if the advanced user wants to verify in details the generated stimuli.

The **Go** button generates all the required simulation templates in subfolder **excitation** (e.g., c:\temp\example1\excitation\), as indicated in the log space in the main window that is automatically updated by the tool and that will help the user to track all the operations that are performed.



As an example, for the HSPICE format, the simulation templates generated by the tool are:

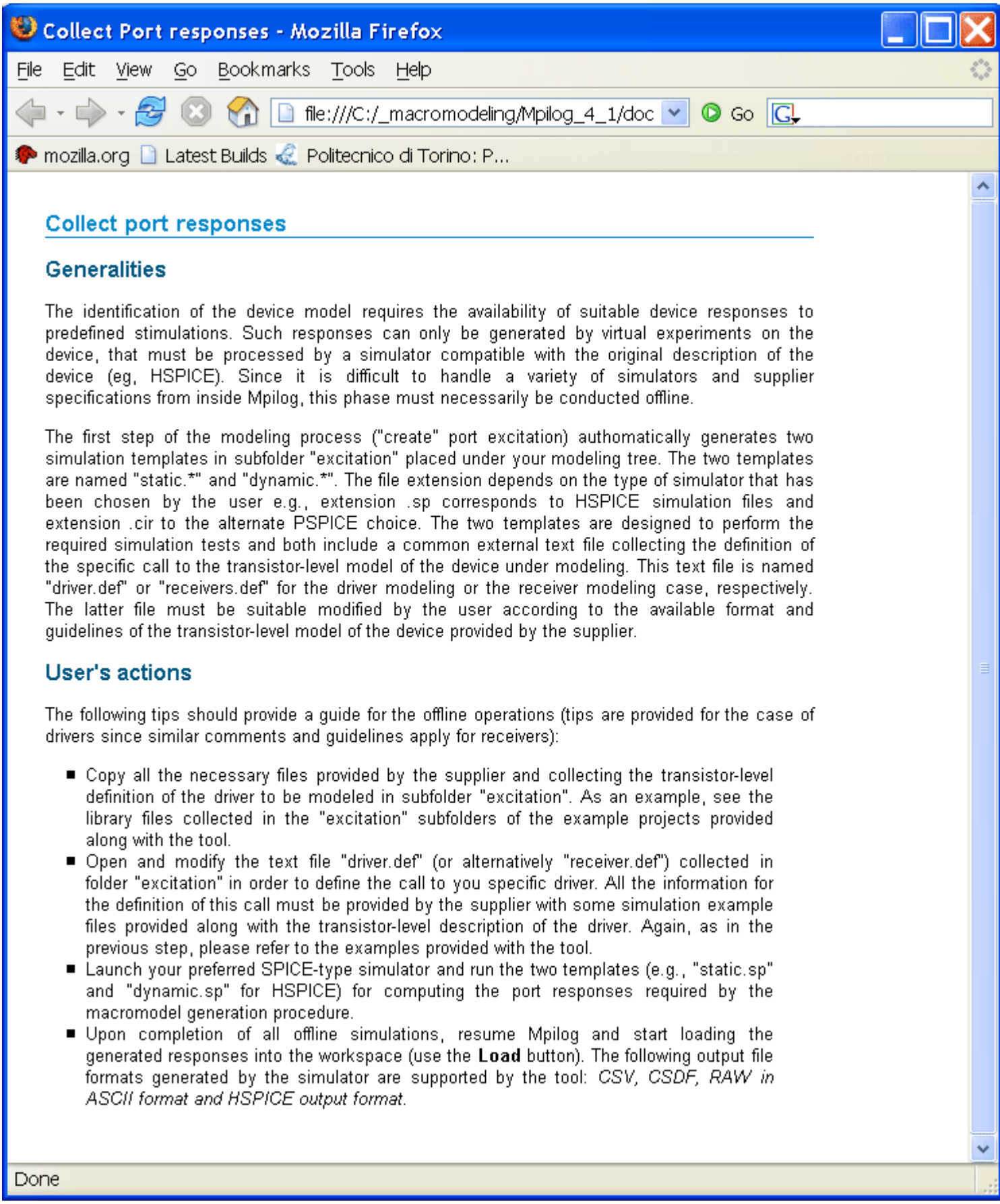
- file "static.sp": template for the computation of the device static characteristics.
- file "dynamic.sp": template for the computation of the transient port responses recorded while the driver is forced in fixed logic states or is driven to perform complete state transitions on suitable loads.

Additional information and theoretical details on the above experiments, are available through the literature (references [1-8] at the end of this document).

STEP 2: Collect port responses

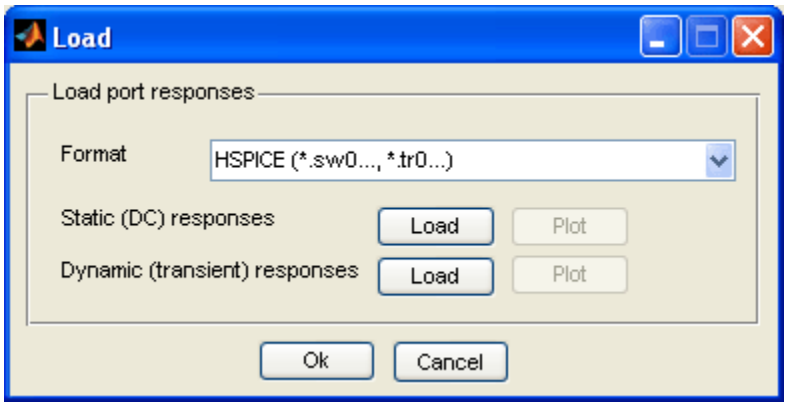
After the templates are generated, the corresponding circuit simulations must be performed, in order to collect all the required responses that allow the computation of the device macromodel. These simulations cannot be run by inside the Mpilog tool, since they are dependent on the specific circuit simulator available to the user.

The **Collect** button instructs the user to go offline and to perform all analyses by means of the preferred circuit simulator, or the one required by the original device description (eg, HSPICE for the example devices). The Collect button opens the following brief html document guiding the user through the off-line simulations.

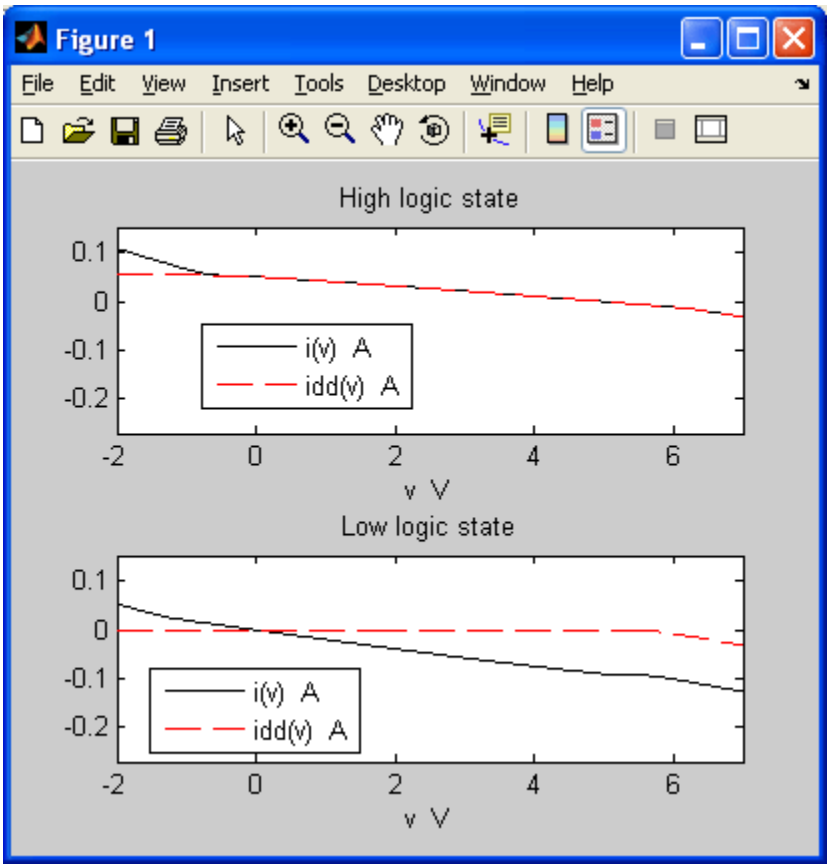

Collect Port responses - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
file:///C:/_macromodeling/Mpilog_4_1/doc
Go
mozilla.org
Latest Builds
Politecnico di Torino: P...
Collect port responses
Generalities
The identification of the device model requires the availability of suitable device responses to predefined stimulations. Such responses can only be generated by virtual experiments on the device, that must be processed by a simulator compatible with the original description of the device (eg, HSPICE). Since it is difficult to handle a variety of simulators and supplier specifications from inside Mpilog, this phase must necessarily be conducted offline.
The first step of the modeling process ("create" port excitation) authomatically generates two simulation templates in subfolder "excitation" placed under your modeling tree. The two templates are named "static.*" and "dynamic.*". The file extension depends on the type of simulator that has been chosen by the user e.g., extension .sp corresponds to HSPICE simulation files and extension .cir to the alternate PSPICE choice. The two templates are designed to perform the required simulation tests and both include a common external text file collecting the definition of the specific call to the transistor-level model of the device under modeling. This text file is named "driver.def" or "receivers.def" for the driver modeling or the receiver modeling case, respectively. The latter file must be suitable modified by the user according to the available format and guidelines of the transistor-level model of the device provided by the supplier.
User's actions
The following tips should provide a guide for the offline operations (tips are provided for the case of drivers since similar comments and guidelines apply for receivers):
Copy all the necessary files provided by the supplier and collecting the transistor-level definition of the driver to be modeled in subfolder "excitation". As an example, see the library files collected in the "excitation" subfolders of the example projects provided along with the tool.
Open and modify the text file "driver.def" (or alternatively "receiver.def") collected in folder "excitation" in order to define the call to you specific driver. All the information for the definition of this call must be provided by the supplier with some simulation example files provided along with the transistor-level description of the driver. Again, as in the previous step, please refer to the examples provided with the tool.
Launch your preferred SPICE-type simulator and run the two templates (e.g., "static.sp" and "dynamic.sp" for HSPICE) for computing the port responses required by the macromodel generation procedure.
Upon completion of all offline simulations, resume Mpilog and start loading the generated responses into the workspace (use the Load button). The following output file formats generated by the simulator are supported by the tool: CSV, CSDF, RAW in ASCII format and HSPICE output format.
Done

STEP 3: Load port responses

Once the simulation templates have been used for computing the device responses via the user's preferred SPICE simulator, the simulation results must be loaded using the **Load** button in the main window, that leads to the follwing window



The user must choose the output format of the simulator (e.g., HSPICE) used to run the simulation templates (e.g., "static.sp" and "dynamic.sp" for HSPICE). The present version of the tool supports the following output file formats: CSV, CSDF, RAW in ASCII format and HSPICE output format. The two **Load** buttons allow the loading of the simulation results. In addition, in order to perform a visual check of the loaded waveforms, the **Plot** buttons, allow the generation of separate plots for all the set of responses. As an example, the first plot button on top, produces the static characteristics of the device, in the High and Low state, on the same graph, as shown below.

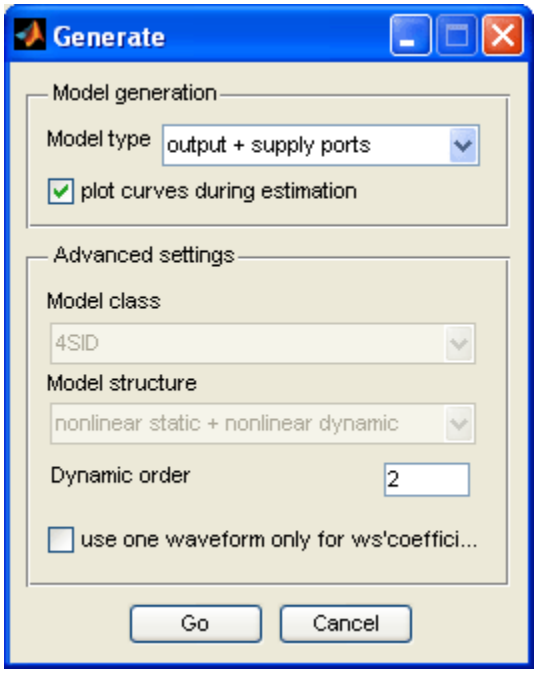


Macromodel generation, validation and implementation

Once the device responses are successfully loaded, Mpilog is ready to identify the device macromodel, to validate it, and perform the implementation, i.e., transfer the results in a script compatible with circuit or hardware description simulation. The user can proceed to one of these steps after the completition of the previous steps.

STEP 4: Macromodel generation

The **Generate** button of Macromodel panel activates the computation algorithm for the identification procedure. Before starting the computation, the user is asked to set few parameters, via the following window:



In a nutshell, this is the meaning of the parameters and possible selections in the above window:

- the **Model type** pupup allows the user to create a macromodel for the output port of the device only or a macromodel including the additional effects of the power supply pin.
- the **Model class** pupup allows the user to chose from the most recent model representations that have been proven to be particularly effective in the estimation of digital device macromodels. When the most effective model representation that has been verified to produce best results for the specific structure at hand (e.g., single ended drivers) is available, the popup might be disabled.
- the **Model structure** defines the structure assumed by the device model. The two different selections in the popup window are model structures defined by a single fully nonlinear dynamic model and model structures consisting of the sum of a static contribution and a dynamic part. Both selections are the common model structures used in literature for the modeling of an unknown system from its external observations. Again, when a specific choice has been proven to provide best results, the popup is disabled.
- the **Maximum model size** is the maximum number of mathematical functions that may be used to build the model; once again, experience tells that a typical model size is within the range [1,8] and there is very little to gain at larger values.
- the flag **use one waveform only...** improves the convergence of the estimation of the model parameters for critical devices exhibiting a stiff behavior during state switching. Specifically, the model representation assumed for digital drivers is a weighted combination of submodels accounting for the driver behavior in fixed High and Low logic states. Since the weights combining the submodels are

obtained from device responses to suitable loads (2 loads are typically used), this flag allows to simplify the computation of the weights by using one load only.

It is also suggested that unexperienced users do not attempt to change the default values, unless in cases of lack of convergence of the procedure. In such cases, the user may want to try to increasing the values of the above parameters one at the time.

Should the user wish to display a comparison between the identification curves and the model responses during the model estimation process, the **Show curves** box may be ticked.

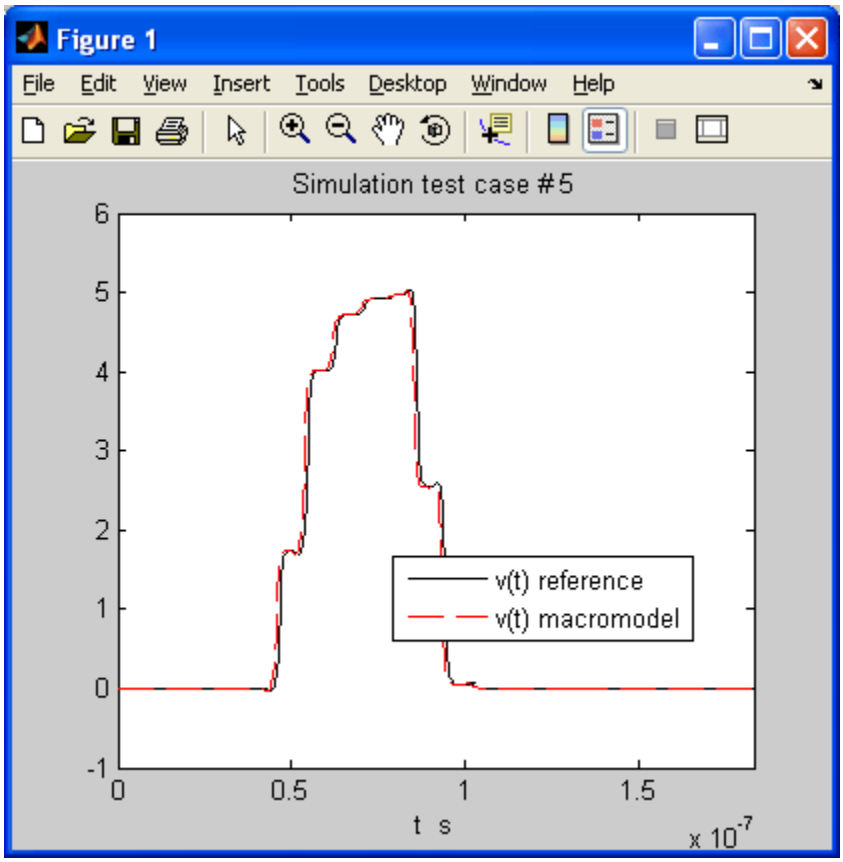
The **Go** button actually starts the generation of macromodel and reports the details of all steps completed during the estimation procedure in a Log window. Details of the structure of port macromodel are reported in references [1-8] at the end of this document.

Upon completion of the macromodel generation, the **Validation** and **Implementation** buttons are enabled, in order to allow the user to check the overall accuracy of the macromodel and to implement the macromodel in the preferred format (SPICE, ASTAP, VERILOG-A,...), respectively.

STEP 5: Macromodel validation

The **Validate** button allows to automatically check the accuracy of the generated model before implementing and using it. The validation is done by computing the macromodel response to a simulation test cases by comparing the model response and the reference one obtained with the transitor-level description of the example device. The simulation test case consists of the example device performing two complete state transitions (bit pattern "010") while its output port is connected to a distributed transmission line and the supply pin is connected to an ideal VDD battery. More details on the test case performed are in the section labeled Test#5 in the template file "dynamic.sp".

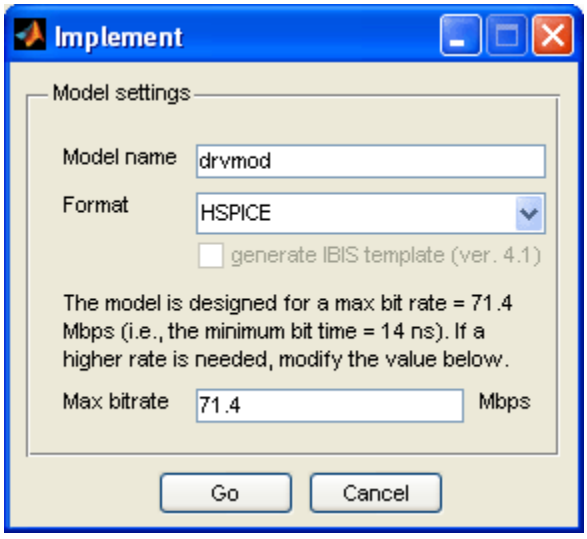
Finally, when all the validation steps are successfully completed, a plot collecting the results of the validation test appear. As en example, the following figure shows the comparison between the reference response (solid line) and the macromodel responses (dashed line) for the validation test.



STEP 6: Macromodel implementation

The **Implement** button takes the user to a new window (see below), where several settings can be chosen, ie

- **Model name** allows the user to define a name for this particular implementation.
- Model **Format** allows the user to select the implementation be compatible with different circuit solvers (HSPICE, PSPICE) or hardware description languages (at present, VERILOG-A is included only).



The **Go** button generates the macromodel implementation as a new file (named by the device name specified in the Implement window). The file is created in folder "c:\temp\example1\macromodel\HSPICE" (or in subfolder PSPICE,... for different implementations). It is suggested that the user moves this file in the workspace of his/her preferred simulator, and starts using it.

3.1 Additional examples

As outlined in the text file "examples.txt" that is available in the compressed folder "examples.zip", different example project workspaces are provided. Since most of these examples are based on devices provided by a commercial supplier, it is higly recommended to carefully read the supplier documentation referring to the simulation examples (please go to the official suppliers websites). Besides, a detailed look into the examples folders storing the previously saved workspaces for these examples, and in particular the subfolder "excitation" under the project tree, is also recommended. Users can follow the same procedure suggested for the first example in this tutorial.

The additional examples included in this distribution are listed below.

- Folder **"didrv_fairchild"** collects the modeling of the output and supply ports of a commercial Fairchild differential driver.

Notes

1. "I/O Buffer Information Specification (IBIS) Ver. 4.1," on the web at http://www.eigroup.org/ibis/, Feb. 2004.

2. Ph.D. Thesis "Behavioral Modeling of Nonlinear Circuit Elements: Application to Signal Integrity and Electromagnetic Compatibility, discussed by I.S. Stievano at Politecnico di Torino, Turin, Italy, March, 2001.

References

[1] I.S.Stievano, F.G.Canavero, I.A.Maio, "Parametric Macromodels of Digital I/O Ports," IEEE Transactions on Advanced Packaging, Vol. 25, No. 2, pp.255-264, May 2002.

[2] I.S.Stievano, I.A.Maio, F.G.Canavero, "Behavioral Models of I/O Ports from Measured Transient Waveforms," IEEE Transactions on Instrumentation and Measurement, Vol. 51, No. 6, pp. 1266-1270, Dec. 2002.

[3] I.S.Stievano, I.A.Maio, F.G.Canavero, "M[pi]log, Macromodeling via Parametric Identification of Logic Gates," IEEE Transactions on Advanced Packaging, Vol. 27, No. 1, pp. 15-23, Feb. 2004.

[4] I. S. Stievano, I. A. Maio, F. G. Canavero, C.Siviero, "Parametric Macromodels of Differential Drivers and Receivers," IEEE Transactions on Advanced Packaging, Vol. 28, No. 2, pp. 189-196, May 2005.

[5] I. S. Stievano, F. G. Canavero, I. A. Maio, "Behavioural Macromodels of Digital IC Receivers for Analog-Mixed Signal Simulations," IEE Electronics Letters, Vol. 41, No. 7, pp. 396-397 March 31, 2005.

[6] I. S. Stievano, I. A. Maio, F. G. Canavero, C.Siviero, "Reliable Eye-Diagram Analysis of Data Links via Device Macromodels," IEEE Transactions on Advanced Packaging, Vol. 29, No. 1, pp. 31-38, Feb. 2006.

[7] I. S. Stievano, I. A. Maio, F. G. Canavero, "On-the-fly estimation of IC macromodels", IEE Electronics Letters, Vol. 42, No. 14, pp. 801-803, 6th July 2006.

[8] I. S. Stievano, C. Siviero, F. Canavero, I. Maio, "Composite local-linear state-space models for the behavioral modeling of digital devices," proc. of 2007 IEEE Instrumentation and Measurement Technology Conference, Warsaw, Poland, May 1-3, 2007.

[9] J. Rabaey, "Digital Integrated Circuits: A Design Perspective", Prentice Hall, 1996.